VaanNila

Menu

Struts 1

Struts 2

Spring

Hibernate

Ant

Log4j

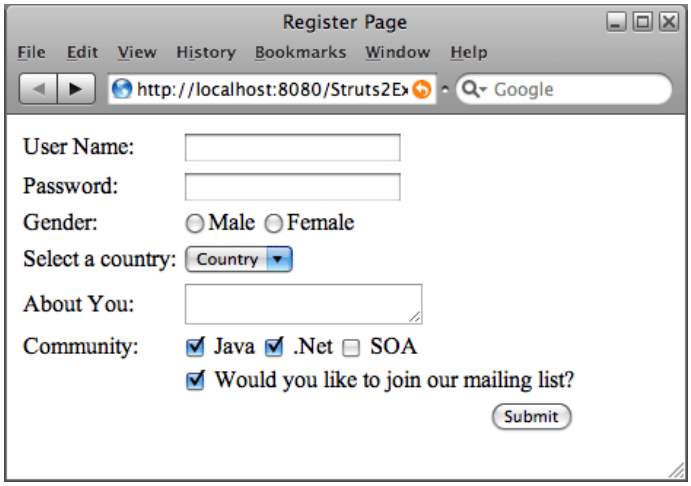
Struts 2 > Struts 2 UI Tags Example

Struts 2 UI Tags Example

Struts 2 UI Tags are simple and easy to use. You need not write any HTML code, the UI tags will automatically generate them for you based on the theme you select. By default the XHTML theme is used. The XHTML theme uses tables to position the form elements.

In this example you wil see how to create a registration page using Struts 2 UI tags. You will also learn how to pre populate the form fields, set default values to it and to retrive the values back in the jsp page.

The register.jsp looks like this



The following code is used to create the register.jsp page

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Register Page</title>
</head>
<body>
<s:form action="Register">
<s:textfield name="userName" label="User Name" />
<s:password name="password" label="Password" />
<s:radio name="gender" label="Gender" list="{'Male','Female'}" />
<s:select name="country" list="countryList" listKey="countryId"
   listValue="countryName" headerKey="0" headerValue="Country"
   label="Select a country" />
<s:textarea name="about" label="About You" />
<s:checkboxlist list="communityList" name="community" label="Community"
   />
<s:checkbox name="mailingList"
label="Would you like to join our mailing list?" />
<s:submit />
</s:form>
</body>
</html>
```

If you view the source of this page you can see the HTML codes generated based on the XHTML theme.

Struts 2 ValueStack

Now lets understand how the UI tags work. In Struts 2 `ValueStack` is the place where the data associated with processing of the request is stored. So all the form properties will be stored on the `ValueStack`. The `name` attribute of the UI tag is the one which links the property on the `ValueStack`.

The next important attribute of the UI tag that you need to understand is the `value` attribute. If you like to populate some default value for a specific field then you need to set that `value` attribute to that value.

The following code will by default set the value of the textfield to "Eswar"

```
<s:textfield name="userName" label="User Name" value="Eswar"/>
```

Here we directly specify the value in the jsp page, instead if you want to set it throught Action then, you can have a property like `defaultName` and set its value to the desired name. In this case the code will look like this.

```
<s:textfield name="userName" label="User Name" value="defaultName"/>
```

Subscribe

RSS

Email

The property `defaultName` is stored on the ValueStack so its value will be set to the textfield. If you think you don't need a seperate form property to do this, then you can set the `userName` property itself to the desired value. In this case you need not specify the `value` attribute seperately. In this example we populate the `community` in this way.

The value set in the `label` attribute of the UI tags will be used to render the label for that particular field while generating the HTML code.

Now lets see the flow of the example. First the index.jsp page will be invoked by the framework.

```
index.jsp
~~~~~~~~~
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=populateRegister.action">
```

Here we forward the request to the `populateRegister` URL. Based on the mapping done in the `struts.xml` file the `populate()` method in the `RegisterAction` class will be called. Here the mapping is done using the dynamic method invocation feature of Struts 2. The struts.xml file contains the following mapping.

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <package name="default" extends="struts-default">
        <action name="*Register" method="{1}"
           class="vaannila.RegisterAction">
            <result name="populate">/register.jsp</result>
            <result name="input">/register.jsp</result>
            <result name="success">/success.jsp</result>
        </action>
    </package>
</struts>
```

The register action class contains the form properties and the corresponding getter and setter methods. It also contains the `execute()` and `populate()` methods. In the populate method we first populate the values and then set the default values for the form fields. The `RegisterAction` class contains the following code.

```
package vaannila;

import java.util.ArrayList;

import com.opensymphony.xwork2.ActionSupport;

public class RegisterAction extends ActionSupport {

    private String userName;

    private String password;

    private String gender;

    private String about;

    private String country;

    private ArrayList<Country> countryList;

    private String[] community;

    private ArrayList<String> communityList;

    private Boolean  mailingList;

    public String populate() {

        countryList = new ArrayList<Country>();
        countryList.add(new Country(1, "India"));
        countryList.add(new Country(2, "USA"));
        countryList.add(new Country(3, "France"));

        communityList = new ArrayList<String>();
        communityList.add("Java");
        communityList.add(".Net");
        communityList.add("SOA");

        community = new String[]{"Java",".Net"};
        mailingList = true;

        return "populate";
    }

    public String execute() {
        return SUCCESS;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
```

```
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getAbout() {
        return about;
    }

    public void setAbout(String about) {
        this.about = about;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public ArrayList<Country> getCountryList() {
        return countryList;
    }

    public void setCountryList(ArrayList<Country> countryList) {
        this.countryList = countryList;
    }

    public String[] getCommunity() {
        return community;
    }

    public void setCommunity(String[] community) {
        this.community = community;
    }

    public ArrayList<String> getCommunityList() {
        return communityList;
    }

    public void setCommunityList(ArrayList<String> communityList) {
        this.communityList = communityList;
    }

    public Boolean getMailingList() {
        return mailingList;
    }

    public void setMailingList(Boolean mailingList) {
        this.mailingList = mailingList;
    }

}
```

Textfield and Password Tags

Now lets see each UI tag in detail. The `textfiled` tag is used to create a textfield and `password` tag is used to create a password field. These tags are simple and uses only the common attributes discussed before.

```
<s:textfield name="userName" label="User Name" />
<s:password name="password" label="Password" />
```

Radio Tag

To create radio buttons we use `radio` tag. The `list` attribute of the `radio` tag is used to specify the option values. The value of the `list` attribute can be a Collection, Map, Array or Iterator. Here we use Array.

```
<s:radio name="gender" label="Gender" list="{'Male','Female'}" />
```

Select Tag

We dispaly the country dropdown using the `select` tag. Here we specify the option values using the `countryList` property of the `RegisterAction` class. The `countryList` is of type ArrayList and contain values of type `Country`. The `Country` class has `countryId` and `countryName` attribute. The `countryName` holds the country value to be display in the frontend and the `countryId` holds the id value to store it in the backend. Here `countryId` is the key and the `countryName` is the value. We specify this in the select tag using the `listKey` and `listValue` attribute. The first value can be specified using the `headerValue` attribute and the corresponding key value is specified using the `headerKey` attribute.

```
<s:select name="country" list="countryList" listKey="countryId"
   listValue="countryName" headerKey="0" headerValue="Country"
label="Select a country" />
```

Textarea Tag

The `textarea` tag is used to create a textarea.

```
<s:textarea name="about" label="About You" />
```

Checkboxlist Tag

The `checkboxlist` tag is similar to that of the `select` tag, the only difference is that it displays boxes for each option instead of a dropdown. It returns an array of String values.

```
<s:checkboxlist list="communityList" name="community" label="Community"
  />
```

Checkbox Tag

The `checkbox` tag returns a boolean value. If the checkbox is checked then true is returned else false is returned.

```
<s:checkbox name="mailingList" label="Would you like to join our mailing
  list?" />
```

Submit Tag

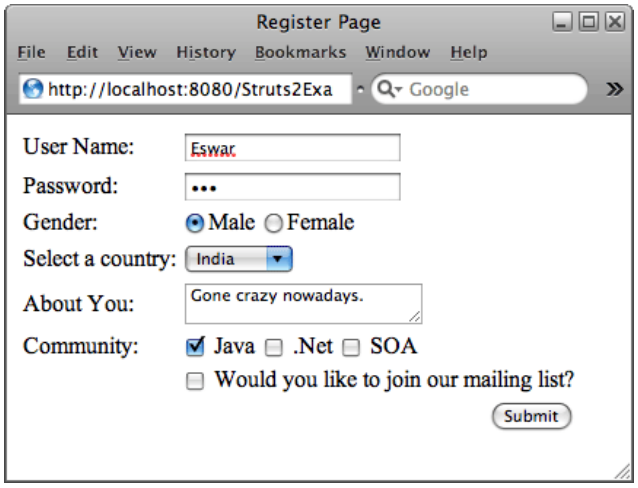
The `submit` tag is used to create the Submit button

```
<s:submit />
```

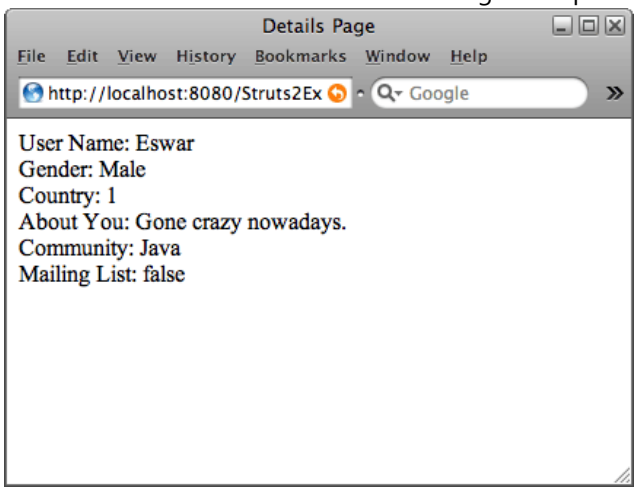
Now lets enter the details and submit the form. The `execute()` method in the `RegisterAction` class will be invoked this time and the user will be forwarded to the `success.jsp` page.

```
success.jsp
-----------
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="/struts-tags"  prefix="s"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Details Page</title>
</head>
<body>
User Name: <s:property value="userName" /><br>
Gender: <s:property value="gender" /><br>
Country: <s:property value="country" /><br>
About You: <s:property value="about" /><br>
Community: <s:property value="community" /><br>
Mailing List: <s:property value="mailingList" />
</body>
</html>
```

Now lets enter the following details and submit the form.



The following registration details will be displayed to the user.



You can download the Struts 2 UI Tags Example by clicking the Download link below.

Source :**Download**

Source + Lib :**Download**